

Scholia 2026: Compliance with SPARQL 1.1

Egon L. Willighagen¹, Daniel Mietchen^{2,3,4}, Peter Patel-Schneider, Konrad Linden⁵, Lars G. Willighagen⁶ and Wolfgang Fahl⁷

¹*Department of Translational Genomics, NUTRIM, Maastricht University, The Netherlands*

²*FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Karlsruhe, Germany*

³*Leibniz Institute of Freshwater Ecology and Inland Fisheries (IGB), Berlin, Germany*

⁴*Institute for Globally Distributed Open Research and Education (IGDORE), Germany*

⁵*Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany*

⁶*Department of Ecology, Radboud Institute for Biological and Environmental Sciences, Radboud University, The Netherlands*

⁷*BITPlan GmbH, Willich, Germany*

Abstract

Scholia is a graphical user interface that uses a combination of SPARQL and the Flask Python platform to visualize data from Wikidata. In this demonstration, we will show how Scholia works and how efforts in the past eighteen months by the Scholia project members make it independent from the Wikidata Query Service Blazegraph installation (WDQS). The reason for this effort was a forced Wikidata graph split in 2025, and most Scholia SPARQL queries were tuned towards the WDQS. The project explored various options, including writing federated SPARQL queries using the new functionalities provided by WDQS, but here, we discuss a solution involving standard SPARQL 1.1 queries, compatible with any SPARQL 1.1 engine, for example QLever.

Keywords

Scholia, Wikidata, SPARQL, Python

Wikidata is a crowd-curated platform providing general reference information in a semantic format [1]. Its official SPARQL service, the Wikidata Query Service (WDQS), has historically relied on the Blazegraph engine to evaluate queries against the graph [2]. However, as Wikidata has grown to over 17 billion triples, Blazegraph has reached significant performance limits. This led the Wikimedia Foundation to split the service into two halves during 2025: WDQS-main, containing roughly 8.6 billion triples of general data, and WDQS-scholarly, containing 8.7 billion triples related to scholarly articles [3].

Scholia [4] is a specialized portal for the scientific community that provides access to this curated content through a set of 387 precomposed SPARQL query templates organized into over 40 profile types, known as aspects. These cover general entities such as authors, works, venues and organizations, as well as taxa or diseases and biochemical aspects like chemicals, genes or proteins [5]. Because Scholia's queries frequently require data from both the general and scholarly subsets of the graph, the split posed a critical challenge to the continuation of the service, and as an interim solution, the Wikimedia Foundation continued to provide a Blazegraph instance with the unified full graph, with a decommissioning date in January 2026. Preparing for this change, the Scholia team investigated three primary recovery routes. The first involved modifying existing queries to use SPARQL federation to communicate between the two split endpoints. This was found to be impractical, as inter-service communication led to severe performance degradation and the loss of optimization opportunities [3]. A second option explored the use of snapQuery middleware [6] to generalize query access across endpoints. The third and ultimately selected route was to remove any Blazegraph-specific constructs in Scholia and instead changing all the queries to standard SPARQL 1.1. This allows the use of any SPARQL 1.1 engine that can effectively evaluate queries against the entire Wikidata graph, including QLever [7], MilleniumDB, and Virtuoso, not only obviating the need to switch to a split-service architecture with its inherent inefficiencies but also leveraging modern engines that are considerably faster than Blazegraph.

SWAT4HCLS 2026: The 17th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences, March 23–26, 2026, Amsterdam, The Netherlands

 [E. L. Willighagen](mailto:egon.willighagen@maastrichtuniversity.nl)

 0000-0001-7542-0286 (E. L. Willighagen); 0000-0001-9488-1870 (D. Mietchen); 0009-0000-7789-7459 (P. Patel-Schneider); 0009-0009-1449-5423 (K. Linden); 0000-0002-4751-4637 (L. G. Willighagen); 0000-0002-0821-6995 (W. Fahl)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

A fork of the original Scholia project was created [8] to collaborate on the necessary modifications, around which several hackathons were organized [9]. A decision was first made to use QLever [7] and a special Wikidata query service that had previously been set up by the QLever team (<https://qlever.dev/api/wikidata>). A benchmark of SPARQL backends [10] showing that QLever was fast on Scholia queries supported this decision.

The bulk of the work was eliminating non-standard constructs from query templates. The major categories of these constructs were: named sub-queries; the Blazegraph locality service that finds nodes with a geographic property value near a given geographic point; the Blazegraph gather-scatter service that finds nodes in the vicinity of a node reachable by a property; the Wikibase service allowing communication with the MediaWiki API for e.g. searching full text contents of Wikipedias; and the Wikibase label service that finds the property value that has the most-preferred language.

These non-standard constructs had largely been used to improve query evaluation. Other systems that use the WDQS also use non-standard constructs to get around problems with query termination caused by the slowness of Blazegraph. Named sub-queries were eliminated by substituting the named query where it was referenced. The locality service was eliminated by unrestricted querying and then filtering based on computed distances. The gather-scatter service was mostly eliminated by using transitive closure and filters or limits.

Perhaps the most problematic construct to eliminate was the Wikibase label service, as it involves preference. To replace the service in standard SPARQL 1.1, a Jinja template was implemented that generates the correct query components. This initial implementation was found to have bad performance when looking up labels for unbound values, potentially requiring a rewrite of the whole query to avoid their encounter. An improvement to the query template was provided that circumvents the performance degradation entirely. As of the end of 2025, the template is only a partial solution for the unavailable label service, until the performance of a complete replacement can be analyzed, and a suitable default label is given when no label is found in any of the requested languages.

Some queries were initially slower under QLever than under Blazegraph. This was unexpected because QLever is generally much faster than Blazegraph. The queries involved OPTIONAL constructs, which when run isolated result in a large subresults. Communications with the QLever team indicated that this was a known, current limitation in QLever that is due to be optimized in early 2026. Several of the problematic queries were refactored to replace their expensive portions by using UNION constructs.

In general, though the process took about eighteen months, the overall results are quite satisfactory. Only very few queries still have problems as of the end of 2025. As several queries were problematic in Blazegraph as well, the few remaining problems with the QLever backend do not indicate a reduction in overall performance of Scholia. Our effort shows that it is generally possible to adjust any system that uses the WDQS to instead use standard SPARQL 1.1 queries.

The next step in updating Scholia to only utilizing standard SPARQL 1.1 queries is to set up an official Wikimedia Foundation Wikidata query service using a SPARQL 1.1 engine. If, contrary to current expectations, that service does not use QLever, the Scholia queries may have to be adjusted to tune them to that engine. Finally, the official Scholia service will have to be updated to use the new QLever-backed version of Scholia.

The demo at SWAT4HCLS will be set up to demonstrate Scholia, along with opportunities to investigate the changes made in the conversion and effects of using QLever.

Acknowledgments

The authors thank everyone who has contributed patches or bug reports to the Scholia project, particularly Finn Nielsen (for the original concept of Scholia), Hannah Bast, Moritz Schubotz and Lane Rasberry. A rich list of contributors to the project is available at <https://github.com/WDscholia/scholia/graphs/contributors>. The work described here was supported in part by the German Research Foundation through the find.software project [11] (funded under grant number 567156310) and by the European Commission through the AQUANAVI initiative [12] of the OSCARS project (101129751).

Declaration on Generative AI

Some author(s) have occasionally employed Generative AI tools for the changes in the Scholia code. All code changes have been reviewed by another person.

References

- [1] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* 57 (2014) 78–85. doi:10.1145/2629489.
- [2] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, A. Bielefeldt, Getting the most out of Wikidata: Semantic technology usage in Wikipedia’s knowledge graph, in: International Semantic Web Conference, Springer, 2018, pp. 376–394.
- [3] T. Lubiana, L. Rasberry, D. Mietchen, The Wikidata Query Service Split and its Impact on the Scholarly Graph, in: CEUR Workshop Proceedings, volume 4064, CEUR-WS.org, 2024. URL: <https://ceur-ws.org/Vol-4064/PD-paper3.pdf>.
- [4] F. Å. Nielsen, D. Mietchen, E. Willighagen, Scholia, Scientometrics and Wikidata, in: The Semantic Web: ESWC 2017 Satellite Events, ESWC 2017 Satellite Events, Springer International Publishing, Portorož, SI, 2017, pp. 237–259. doi:10.1007/978-3-319-70407-4_36.
- [5] E. Willighagen, D. Slenter, A. Rutz, D. Mietchen, F. Å. Nielsen, Scholia Chemistry: access to chemistry in Wikidata (2025). doi:10.26434/CHEMRXIV-2025-53NOW.
- [6] W. Fahl, T. Holzheim, C. T. Hoyt, D. Priskorn, E. Willighagen, SnapQuery, 2025. URL: <https://github.com/WolfgangFahl/snapquery>.
- [7] H. Bast, B. Buchhold, QLever: A Query Engine for Efficient SPARQL+Text Search, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM ’17, Association for Computing Machinery, New York, NY, USA, 2017, p. 647–656. doi:10.1145/3132847.3132921.
- [8] F. Å. Nielsen, D. Mietchen, E. Willighagen, H. Bast, Scholia, 2017. URL: <https://github.com/ad-freiburg/scholia/tree/qlever>, GitHub repository qlever branch.
- [9] Wikidata, Scholia Events December 2025, 2025. URL: https://www.wikidata.org/wiki/Wikidata:Scholia/Events/2025_12.
- [10] P. F. Patel-Schneider, Benchmarking SPARQL Engines on Wikidata Queries, in: CEUR Workshop Proceedings, volume 4108, CEUR-WS.org, 2024. URL: <https://ceur-ws.org/Vol-4108/paper3.pdf>.
- [11] R. Gey, D. Mietchen, O. Karras, T. Wittenborg, M. Schubotz, J. Bumberger, find.software: Foundations for Interdisciplinary Discovery of (Research) Software, *Research Ideas and Outcomes* 11 (2025) e179253. doi:10.3897/rio.11.e179253.
- [12] T. Heger, S. Angela Berger, J. M. Jeschke, C. Kittel, P. Kraker, A. K. Makower, D. Mietchen, J. C. Nejstgaard, M. Schramm, AQUANAVI: Navigating Grand Challenges and their Mitigation using Aquatic Experimental RIIs, *Research Ideas and Outcomes* 11 (2025) e176476. doi:10.3897/rio.11.e176476.